บุญเลิศ เอี่ยมทัศนา

เทอร์โบปาสคาลกับวินโดวส์ : ไอคอนของขวัญ

วิธีการสร้างไอคอน และภาพกราฟิกแบบบิตแม็พ พร้อมกับเคอร์เซอร์ ส่วนตัวส่วนตัว

ผู้ที่ใช้งานวินโดวส์มักจะมีไฟล์ข้อมูลเป็นส่วนตัว เช่นอาจเป็นไดอารีหรือบันทึก และมักจะทำเป็นโปรแกรมไอเท็มไอคอนไว้ ที่ใช้กันคือ ยืมไอคอนมาจากแอพพลิเคชันอื่น ใครคนนั้นอาจคอยไอคอนส่วนตัวของขวัญปีใหม่จากคุณ ของขวัญชิ้นเล็กที่มากด้วยน้ำใจ

ผมจึงขอสวัสดีปีเก่าด้วย ไอคอน และเลยไปถึงบิตแม็พเผื่อจะเป็นการ์ด ส.ค.ส. อิเล็กทรอนิกส์ ปิดท้ายด้วยเคอร์เซอร์สำหรับตัวคุณเองครับ

ไอคอน

การสร้างไอคอนไม่ใช่เรื่องยาก หากมี WRT (Whitewater Resource Toolkit) เพราะเพียงแต่เปิดใช้ WRT เลือกปุ่ม Icon จะได้ไอคอนเอดิเตอร์มีเครื่องมือพร้อมสำหรับการวาดภาพไอคอนดังรูปที่ 1

แต่รูปนี้ผมเก็บเมื่อได้วาดภาพแล้ว เพื่อที่จะใช้ประกอบคำอธิบายต่อไปหากเป็นเมื่อเริ่มต้นแท้ ๆ จะไม่มีกรอบสี่เหลี่ยม 2 กรอบที่มีภาพวงกลมอันเกิดจากการวาด พื้นที่ตรงนี้จะว่างส่วนกรอบสี่เหลี่ยมทั้งสองนี้กรอบซ้ายคือ กรอบวาด กรอบขวาคือ กรอบแสดงผล

ขั้นต้น

เริ่มการวาดภาพได้ด้วยการเลือกรายการ File/New เมื่อเป็นภาพใหม่ ถ้าเป็นภาพเก่าที่จะนำมาตบแต่งให้เลือกรายการ File/Open

จะแสดงไดอะล็อกบ็อกซ์ดังรูปที่ 2 ให้เลือกจำนวนสีตามจอภาพที่ใช้ มีให้เลือก 3 ชนิดด้วยกันคือ 16, 8 และ 2 (8 สำหรับ EGA 2 สำหรับโมโน) ซึ่งในการทำไอคอนหากต้องการให้ใช้งานบนเครื่องใดก็ได้ จะต้องสร้างทั้งสามชนิดนี้ให้ชื่อรีซอร์สชื่อเดียวกัน

โดยที่วินโดวส์จะเป็นผู้เลือกรีซอร์สที่เหมาะสมกับเครื่อง แต่ถ้าต้องการใช้กับจอวีจีเอ ทั้งสีหรือโมโนโครมแน่ ๆ สร้างเพียงชนิดเดียวก็พอคือ 16 สี สำหรับขนาดคงมีขนาดเดียวเท่ากันหมดคือ 32×32 จุดภาพ

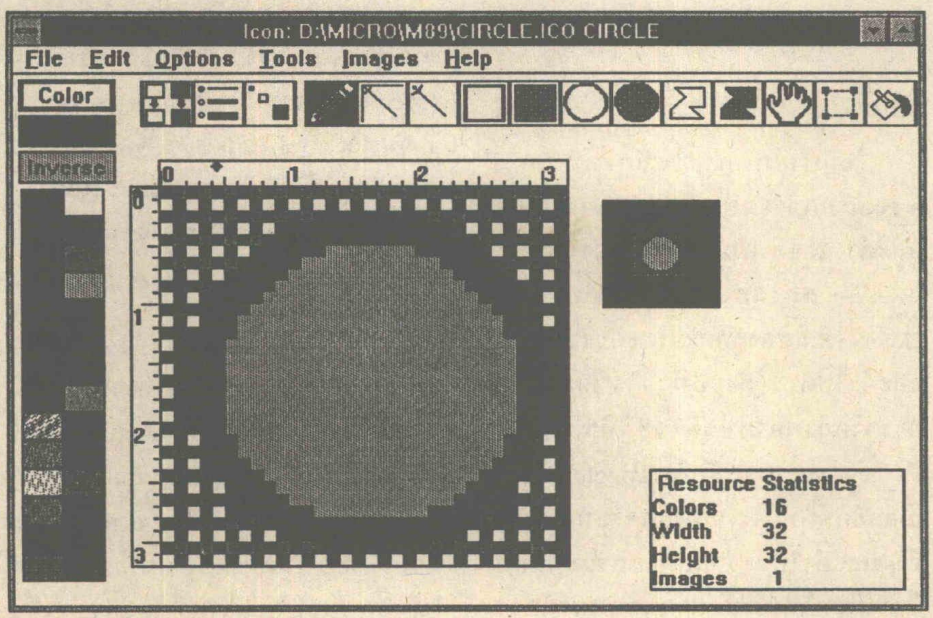
กรอบวาดและกรอบแสดงผล

เมื่อเลือกจำนวนสีแล้วจะแสดงกรอบวาดและกรอบแสดงผลดังแสดงในรูปที่ 1 กรอบวาดเป็นกรอบที่ใช้ในการวาด ซึ่งจะแสดงผลเท่าของจริงในกรอบแสดงผล โดยแสดงตามทันทีเช่น เมื่อลากเส้นในกรอบวาดก็จะปรากฏเส้นในกรอบแสดงผลตามไปด้วย

จานสี (Pallette)

ในการวาดภาพจะต้องใช้ทั้งสีและเครื่องมือในการวาดภาพ โดยจะขอกล่าวถึงเรื่องสีก่อนคือ ทางด้านซ้ายของรูปที่ 1 จะเห็นเป็นตารางสี่เหลี่ยมเรียกว่าจานสี มีอยู่ด้วยกัน 28 ช่อง (หากสงสัยในเรื่องบอดสี ทดสอบได้ครับจะต้องเห็นทั้งหมด 28 สีบนจอวีจีเอสี) แบ่งเป็น

- สีเดี่ยว สี 8 คู่บนเป็นสี่เดี่ยว คือสีที่จุดภาพทุกจุดเป็นสีเดียวกันหมด
- สีผสม อีก 6 คู่ล่างเป็นสีผสม คือสีที่จุดภาพแต่ละจุดไม่เหมือนกัน (2 ถึง 3 สี) จะเห็นฉนัดเมื่อระบายเป็นสีพื้นดังในรูปที่ 1



รูปที่ 1 หน้าต่างไอคอนเอดิเตอร์ของ WRT

ชนิดของสี

เหนือจานสีขึ้นไปมีกรอบสีอีก 3 กรอบเป็นกรอบชนิดของสี ซึ่งผมขอกล่าวก่อนว่าในการวาดภาพจะต้องเลือกสีที่จะใช้วาดดังที่กล่าวมา เครื่องมือตามที่จะกล่าวในหัวข้อต่อไป และชนิดของสีตามที่จะกล่าวในหัวข้อนี้ ซึ่งสามสิ่งนี้จะเลือกสิ่งใดก่อนและจะเลือกกลับไปกลับมาอย่างไรก็ได้

สำหรับกรอบชนิดของสีทั้งสามกรอบนี้มีเรื่องที่เหมือนกันและสัมพันธ์กันคือ เมื่อคุณเลือก (คลิ๊ก) กรอบใดจะทำให้กรอบนั้นแอคตีฟและทำให้อีกสองกรอบไม่แอคตีฟ และถ้าไปเลือกสีในจานสี จะทำให้สีในกรอบที่แอคตีฟเปลี่ยนตาม เท่ากับเป็นการเลือกสีเพื่อการวาดตามชนิดของสี คือ

- สีปกติ (Color) เลือกกรอบนี้คือ กรอบบน เพื่อการวาดภาพตามปกติ
- สีพื้นจอภาพ (Screen) หากตรงส่วนใดของภาพที่ต้องการให้เป็นสีเดียวกับพื้นจอภาพ ให้เลือกกรอบนี้คือกรอบที่อยู่ตรงกลาง (หากสงสัยสักครู่ครับ)
- สีพื้นจอภาพกลับ (Inverse) คือกรอบล่าง เช่นเดียวกับสีพื้นจอภาพแต่จะได้สีเป็นสีกลับของสีพื้นจอภาพ เช่น ดำเป็นขาว ชมพูม่วงเป็นเขียว และถ้าเปลี่ยนสีในกรอบใด อีกกรอบหนึ่งจะเปลี่ยนตามเป็นสีกลับของกันและกัน

ทั้งสามเรื่องนี้ผมงงอยู่นานจนต้องสร้างไอคอนเขียนโปรแกรมนำไปใช้จึงเข้าใจ ซึ่งผมจะขอกล่าวเฉพาะสีพื้นจอภาพกับสีกลับของมัน คือ

- เรื่องแรกสมมติว่า คุณใช้สีเขียวเป็นสีพื้นจอภาพ เมื่อนำไปวาดซึ่งส่วนมากเป็นการระบาย ดังเช่นในรูปที่ 1 ซึ่งผมใช้สีปกติเป็นสีฟ้าเทา หาก

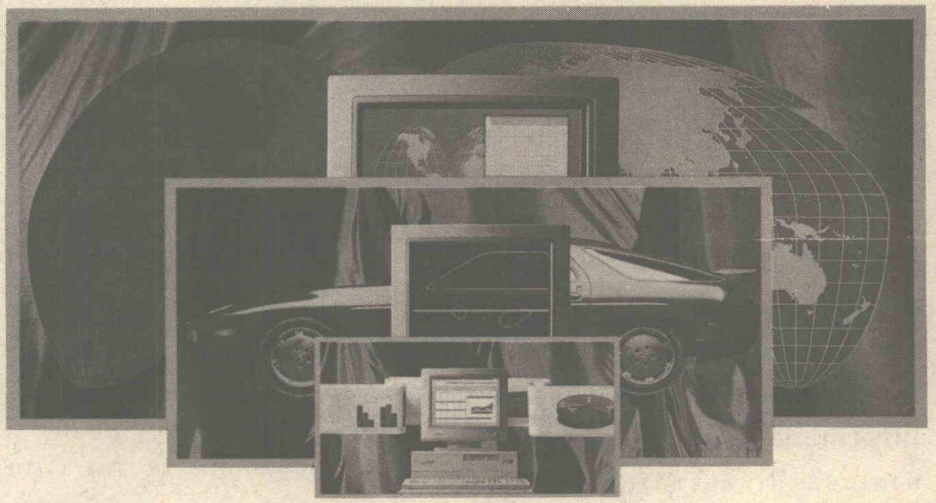
เปลี่ยนเป็นใช้สีพื้นจอภาพเป็นสีเขียว ก็จะได้ภาพที่ระบายเป็นสีเขียว และถ้ากลับไปเลือกสีในจานสีใหม่ให้เป็นสีดำ ภาพที่จะระบายไว้ก็จะเปลี่ยนเป็นสีดำด้วย

– แต่เมื่อทำเป็นไอคอน เมื่ออยู่ในหน้าต่างกลุ่มพื้นที่ที่ระบายไว้ด้วยสีเขียว (หรือจะเป็นสีอื่นใดก็ตาม) จะกลายเป็นสีขาว คือสีพื้นของหน้าต่างกลุ่ม และเมื่อรันแล้วมินิไมส์เป็นไอคอนงาน พื้นที่ระบายไว้ด้วยสีเขียวจะกลายเป็นสีเทาตามพื้นจอภาพ (ตรงที่ไม่มีหน้าต่างของวินโดวส์ 3.1) ทำให้เห็นเป็นภาพไอคอนลอยอยู่ เช่นตามรูปที่ 1 จะเห็นเป็นวงกลมแดงไม่มีกรอบ

– หากนำไอคอนเข้ามาในหน้าต่างงาน เช่นของไอคอนเอดิเตอร์ สีพื้นคือสีเขียวเดิมจะไม่เปลี่ยนตามคงเป็นสีเทาดังที่เมื่ออยู่บนพื้นจอภาพ ทำให้เห็นกรอบของไอคอนถนัด

– นั่นคือ สีพื้นจอภาพที่ให้เปลี่ยนได้ก็เพื่อให้เทียบว่าหากสีพื้นของจอภาพจริงเป็นเช่นใด จะได้ภาพเป็นอย่างไร (สีพื้นจอภาพเปลี่ยนได้ตามความต้องการของผู้ใช้)

– สำหรับสีพื้นจอภาพกลับ จะตรงตามตัวคือ เมื่อสีพื้นเป็นสีเทาตัวเองจะเป็นสีเทาเข้ม เมื่อสีพื้นเป็นสีขาวตัวเองจะเป็นสีดำ ส่วนมากจึงใช้เป็นขอบของภาพที่ต้องการระบายด้วยสีอ่อน ที่เป็นพิเศษคือเมื่อเลื่อนไอคอนดังเช่นในรูปที่ 1 ผมระบายสีนี้ไว้ในวงกลมแดง เมื่อเลื่อนไอคอนจะเห็นภาพข้างหลังเป็นสีกลับ เช่นตัวอักษรดำบนพื้นขาวจะเห็นเป็นขาวบนพื้นดำ คือทำให้ภาพข้างหลังที่เป็นของหน้าต่างงานเป็นสีกลับ แต่เมื่อปล่อยมือจะกลายเป็นสีกลับของสีพื้นจอภาพ



เครื่องมือ

เครื่องมือที่ใช้ในการวาดภาพได้ทำเป็นปุ่มควบคุมไว้ใต้เมนูบาร์ และมีรายการให้เลือกตามปุ่มนี้ในเมนูด้วย โดยแบ่งออกเป็น 2 กลุ่มคือ สามปุ่มแรกอยู่ใน Option ที่เหลืออยู่ใน Tools จะขอกล่าวตามลำดับจากซ้ายไปขวา และจะนำคำในภาษาอังกฤษที่เป็นรายการในเมนูมากำกับไว้ด้วย คือ

กลับสี

อยู่ซ้ายสุดภายในมีกรอบเล็ก ๆ ซึ่งขอเรียกว่าจุดสีอยู่ 4 จุด ขาว 2 จุด และดำ 2 จุด เป็นปุ่มสองสถานะคือ

– สถานะปกติ (Draw with Color) จะแสดงลูกศรจากจุดขาวไปขาว และดำไปดำ (ดังรูปที่ 1) ใช้สำหรับการวาดภาพตามปกติด้วยสีที่เลือกไว้

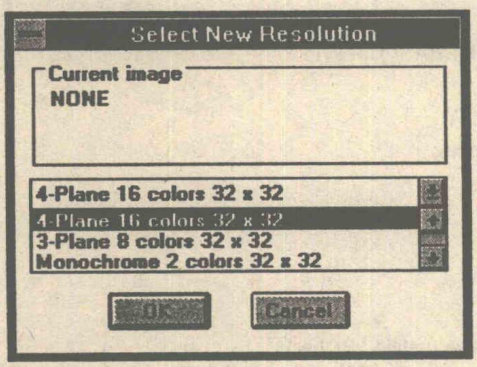
– สถานะกลับสี (Toggle Color) เมื่อคลิ๊กที่ปุ่มนี้จะแสดงลูกศรจากขาวไปดำ และดำไปขาว แสดงสถานะกลับสี คือ เมื่อคลิ๊กเม้าส์ หากคลิ๊กบนสีใดจะได้สีกลับของสีพื้นในการวาดจนกว่าจะปล่อยมือจากเม้าส์ เช่น มีพื้นสีขาวและดำอยู่ติดกัน หากกคลิ๊กเม้าส์ในพื้นสีขาวแล้วลากเส้นจะได้เส้นสีดำถึงแม้จะลากเลยเข้าไปในพื้นสีดำ ต่อเมื่อปล่อยมือแล้วคลิ๊กใหม่จึงจะได้สีขาว ใช้ได้กับการวาดภาพด้วยดินสอเท่านั้น

ขนาดของเส้น

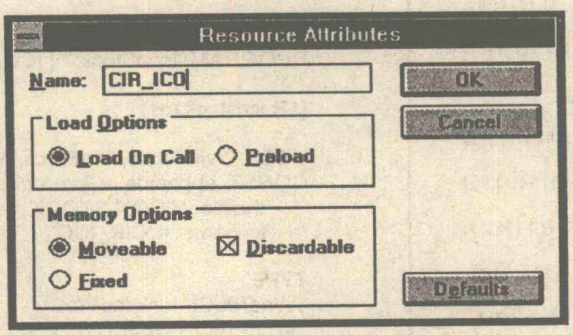
แสดงขนาดของเส้นให้เลือก 3 ขนาด คือ บาง (Thin) ปานกลาง (Medium) และหนา (Thick) คือขนาด 1 จุด 3×3 จุด และ 5×5 จุดตามลำดับ การเลือกให้คลิ๊กวนจนกว่าจะได้ขนาดที่ต้องการ เมื่อเลือกขนาดใดจะแสดงจุดดำที่ขนาดนั้น

ขยายภาพ

เพื่อการกำหนดขนาดของกรอบวาดให้เป็นเท่าของจริง (Actual size) ขยาย 4 เท่า (Magnify four times) ขยาย 8 เท่า (Magnify eigh times) ตามรูปที่ 1 ให้ขยาย 8 เท่า การเลือกให้คลิ๊กวนจนได้ขนาดที่ต้องการ ซึ่งกรอบวาดจะเปลี่ยนขนาดตาม



รูปที่ 2 ไดอะล็อกบ็อกซ์ให้เลือกจำนวนสี



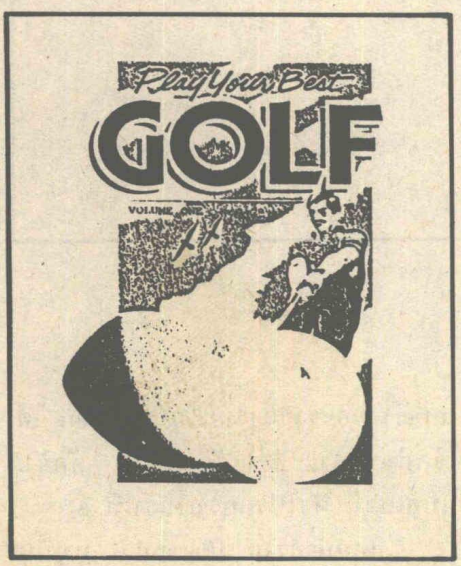
รูปที่ 3
ไดอะล็อกบ็อกซ์ให้ป้อนค่ารายละเอียดของรีซอร์ส ค่าที่แนะนำไว้เว้นชื่อ (ซึ่งต้องป้อน) เป็นค่ามาตรฐาน คงเลือกใช้ตามนี้

มือเลื่อน (Drag)

ใช้เลื่อนผืนภาพทั้งผืนเช่นเดียวกับที่ใช้มือเลื่อนกระดาษโดยคลิ๊กในกรอบวาดแล้วเลื่อนไปในทิศทางที่ต้องการให้ผืนภาพเลื่อน (คือแทนที่จะใช้สโครลบาร์) แต่ใช้ได้กับบิตแม็พเท่านั้น

ที่ตัดภาพ (Select)

ใช้ในการตัดภาพเพื่อการลอก (copy) หรือตัด (cut) ให้แก่คลิปบอร์ด ส่วนมากใช้กับบิตแม็พ



เก็บลงไฟล์

เมื่อวาดภาพไอคอนเสร็จแล้วก็ต้องเก็บลงไฟล์ครับ เนื่องจากเป็นภาพแรกให้เลือกรายการ File/Save As จะถามชื่อไฟล์ผมป้อนเป็น ICOSTOCK.RES ก็คงตามแต่จะตั้งกันครับ ที่ให้นามสกุลเป็น RES ก็เพื่อให้เป็นไฟล์ของรีซอร์ส จะได้อ่านมาใช้ในโปรแกรมได้โดยตรง หากให้เป็น ICO จะเก็บไอคอนเป็นแต่ละภาพ

เมื่อป้อนชื่อไฟล์แล้วจะมีไดอะล็อกบ็อกซ์ให้ป้อนชื่อรีซอร์สของไอคอนนี้ ดังรูปที่ 3 ซึ่งผมป้อนเป็น CIR_ICO ควรจะหาชื่อเพราะ ๆ มาตั้งครับ และคงต้องวาดภาพไอคอนต่อ ๆ ไปสำหรับปีใหม่ ไอคอนหลัง ๆ นี้ให้เก็บลงไฟล์เดียวกันโดยใช้ File/Save Into

ฝากไว้ในโปรแกรม

เช่นเดียวกับการ์ด ส.ค.ส. จะต้องหาซองใส่ ซองอิเล็กทรอนิกส์ของเราทำได้ไม่ยากครับ ดังโปรแกรมในรูปที่ 4 ซึ่งผมตั้งชื่อไฟล์เป็น FORYOU.

```
PROGRAM ForYou;

{$R icostock.res}

USES WObjects;

CONST
  appname = 'foryou';

VAR  app : TApplication;

BEGIN
  app.Init(appname);
  app.Run;
  app.Done;
END.
```

รูปที่ 4 โปรแกรม Foryou

PAS ก็แล้วแต่จะตั้งตรงที่ใช้ไดเรคตีฟ R ระบุชื่อไฟล์ ICOSTOCK.RES คือตรงที่นำไอคอนมาไว้ในโปรแกรม

จากนี้ให้คอมพายล์เป็นไฟล์ .EXE ในการคอมพายล์หากไม่ได้จัดไดเรคทอรีสำหรับไฟล์ .RES ไว้ จะต้องให้ไฟล์ทั้งสองนี้อยู่ในไดเรคทอรีเดียวกัน ได้ไฟล์ FORYOU.EXE สำหรับใครคนนั้น

กับแอพพลิเคชัน

โปรแกรม Foryou คงต้องเรียกว่าเป็นแอพพลิเคชันที่ประหลาดในตัวมีไอคอนมากมาย (ตามที่คุณสร้างไว้)

แต่ตัวเองไม่ได้ใช้เลย หากมินิไมส์จะแสดงไอคอนเหมือนกันแต่เป็นไอคอนของระบบ

หากต้องการให้ใช้ไอคอนที่ตนมีอยู่ ต้องเขียนโปรแกรมต่อดังโปรแกรม IconStock ในรูปที่ 5 คือ ต้องโหลดไอคอนในโปรแกรมของกิจกรรม GetWindowClass ของหน้าต่างคือ WndProC สำหรับ Init หากเขียนโปรแกรมเท่านี้ไม่จำเป็นต้องมี ที่ผมกำหนดไว้เพื่อจะต่อเติมโปรแกรมต่อไป

แสดงไอคอนในหน้าต่างงาน

เจตนาของผมที่ให้ชื่อโปรแกรมว่า IconStock เพื่อเป็นที่สำหรับตุนไอคอนไว้ เมื่อมีโอกาสก็หาไอคอนที่ชอบใจมาเก็บไว้ และเมื่อต้องการใช้จะได้ก๊อบปี้ด้วย WRT จากไฟล์ ICOSTOCK.RES ไปให้แก่ไฟล์ .RES ของแอพพลิเคชันใหม่

แต่การค้นหาไอคอนโดยให้แสดงภาพด้วย WRT เป็นวิธีที่ไม่สะดวกอย่างยิ่งผมจึงจะให้แสดงภาพไอคอนในหน้าต่างของ IconStock คล้ายกับที่เห็นในหน้าต่างกลุ่มเช่นใน Main

วิธีการ

วิธีการคือ เมื่อเราสร้างหรือก๊อบปี้ไอคอนจะต้องจดชื่อรีซอร์สของไอคอนนั้นอยู่แล้ว ผมจะนำมาทำเป็นชื่อรายการในเมนู เมื่อผู้ใช้เลือกรายการใดก็จะอ่านชื่อรายการนั้นมาโหลดไอคอนออกแสดง

ดังเมนูในรูปที่ 6 และผลการรันในรูปที่ 7 ภาพไอคอนที่มีหมายเลขผมก๊อบปี้จากไฟล์ MORICONS.DLL ของวินโดวส์ 3.1 มา 8 รีซอร์ส ซึ่งให้ชื่อรีซอร์สไว้เป็น 1 ถึง 8 ส่วนคำว่า Icon ผมมาเติมในโปรแกรมเพื่อไม่ให้ตัวเลขเหงาครับ

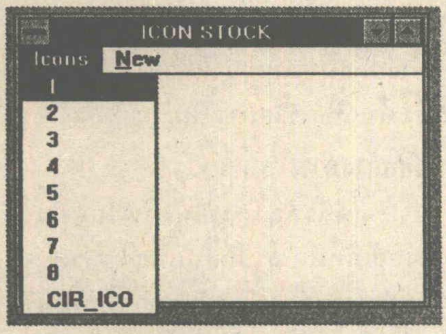
```
PROGRAM IconStock;  { ICOSTCK1.PAS }

{$R icostock.res}

USES  WinTypes,WinProcs,WObjects;
CONST appname  = 'icostock';
      thetitle = 'ICON STOCK';
      iconame  = 'CIR_ICO';

TYPE
 WndProCP = ^WndProC;
 WndProC = OBJECT(TWindow)
 constructor Init(apar : PWindowsObject; atitle : PChar);
 procedure  GetWindowClass(var awndclass : TWndClass); virtual;
         END;

(* ---------- WndProC  Method ---------- *)

CONSTRUCTOR  WndProC.Init(apar : PWindowsObject; atitle : PChar);
BEGIN  TWindow.Init(apar,atitle)  END;


PROCEDURE  WndProc.GetWindowClass(var awndclass : TWndClass);
BEGIN
 TWindow.GetWindowClass(awndclass);
 awndclass.HIcon := LoadIcon(HInstance, iconame);
END;


(* ---------- TheAppC Class ---------- *)

TYPE
 TheAppC = OBJECT(TApplication)
  procedure  InitMainWindow; virtual;
         END;


PROCEDURE  TheAppC.InitMainWindow;
BEGIN
 MainWindow := New(WndProCP, Init(nil,thetitle));
END;

(* ---------- MAIN PROGRAM ---------- *)

VAR  app :  TheAppC;

BEGIN
 app.Init(appname);
 app.Run;
 app.Done;
END.
```

รูปที่ 5 โปรแกรม IconStock

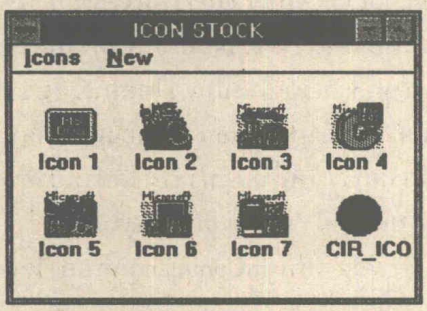
ดักรับเมสเสจ

เมื่อผู้ใช้เลือกรายการ ในเมนูวินโดวส์จะส่งคอมมานด์เมสเสจ คือ wm_Command มาให้ มีค่าไอดี (identifier) ของรายการอยู่ใน wParam ได้จัดกิจกรรม wmCommand ดักรับเมสเสจนี้ ดังโปรแกรมในรูปที่ 8

โดยนำค่าใน lParamLo มาเป็น



รูปที่ 6 เมนูของโปรแกรม IconStock



รูปที่ 7 ผลการรันของโปรแกรม IconStock หลังจากที่ได้เลือกภาพไอคอนในเมนู 8 ภาพ

เงื่อนไข คือจะเป็น 0 หากเป็นเมสเสจที่เกิดจากการเลือกรายการในเมนู ส่งให้ DispIcon นำไปแสดงภาพ ที่เหลือนอกนั้นซึ่งอาจเป็นเมสเสจอันเกิดจากการกดคีย์ลัด (accelerator) หรือจากไดอะล็อกบ็อกซ์ ส่งให้ TWindow.wmCommand รีบไปดำเนินการ

ซึ่งในบางงานอาจให้ TWindow.wmCommand ดักรับคอมมานด์เมสเสจทั้งหมด (คือไม่จัดกิจกรรมนี้ในโปรแกรมของเรา) ซึ่งจะแจงค่าใน wParam ออกเป็นเมสเสจฐานต่าง ๆ ของเมนู คือฐาน cm_First แต่สำหรับโปรแกรมนี้ไม่ต้องการให้แจงค่าในเมสเสจอันเกิดจากการเลือกรายการไอคอนจึงจัดกิจกรรม wmCommand ดักรับเสียเอง แต่ถึงกระนั้นก็ยังต้องการให้แจงค่าอันเกิดจากการเลือกรายการอื่น ๆ จึงส่งให้ TWindow.wmCommand ดำเนินการ

```
PROCEDURE WndProc.wmCommand(var msg: TMessage);
BEGIN
 if (msg.lParamLo = 0) and (msg.WParam < cm_Omenus) then
   DispIcon(msg.WParam)
 else TWindow.wmCommand(msg);
END;


PROCEDURE WndProc.DispIcon(iconId : Integer);
VAR  dc : HDC;  hIcons : HMenu;  tem : String;
     capt : array[0..19] of Char;  iden,res : Word;
BEGIN
 dc := GetDC(HWindow);
 if ox+32 > attr.W then
   begin ox := 20;  oy := oy + 60  end;
 hIcons := GetSubMenu(attr.menu, 0);
 GetMenuString(hIcons, iconId, @capt, SizeOf(capt), mf_ByCommand);
 Val(capt, iden, res);
 if res = 0 then
   begin
   DrawIcon(dc, ox, oy, LoadIcon(HInstance, PChar(iden)));
   tem := 'Icon ' + StrPas(capt);   StrPcopy(capt, tem);
   end
 else
   DrawIcon(dc, ox, oy, LoadIcon(HInstance, @capt));
 TextOut(dc, ox-4,oy+32, @capt, lstrlen(@capt));
 ReleaseDC(HWindow,dc);  ox := ox + 60;
END;
```

รูปที่ 8 โปรแกรมดักรับเมสเสจ และนำภาพไอคอนออกแสดง

โดยกำหนดไว้ว่าค่าซีเอ็มของรายการไอคอนมีค่าได้ตั้งแต่ 1 ถึง 999

นำไอคอนออกแสดง

DispIcon จะนำไอคอนตามค่าไอดีของเมนูออกแสดงครั้งละ 1 ไอคอน (คือตามที่ผู้ใช้เลือก) โดยมีการดำเนินการดังนี้

- จัดค่า ox และ oy เพื่อให้แสดงภาพไอคอนเรียงไปทางขวาและขึ้นบรรทัดใหม่
- อ่านชื่อรายการจากเมนูมาใช้เป็นชื่อไอคอนรีซอร์ส
- โดยที่การตั้งชื่อรีซอร์สอาจตั้งได้ทั้งที่เป็นตัวอักษรและตัวเลข จึงต้องดำเนินการแยกกัน
- ให้แสดงภาพไอคอนในหน้าต่างด้วย DrawIcon และให้แสดงชื่อไอคอนรีซอร์สกำกับไว้ด้วย เพื่อที่จะให้นำไปใช้ในการก๊อบปี้รีซอร์ส

จะเห็นว่าผมไม่ได้กล่าวถึงค่าไอดีของรายการในเมนูเลยว่า เมื่อผมจัดทำเมนูรีซอร์สได้ให้ค่าไว้อย่างไร เพราะว่าจะให้ค่าเท่าไรก็ได้จาก 1 ถึง 999 ตามที่ได้แบ่งค่าไว้ให้

ลบภาพเก่า

โปรแกรมตามนี้หากผู้ใช้เปลี่ยนขนาดของหน้าต่างใหม่หรือถูกซ้อนทับภาพไอคอนจะหายไป ผมจึงให้ค่าเริ่มต้นใหม่แก่ ox และ oy ไว้ใน Paint ในรูปที่ 9 ซึ่งเมื่อมีการเปลี่ยนแปลงขนาดของหน้าต่างจะมีการเรียกมายัง Paint

เช่นกันหากผู้ใช้ต้องการลบภาพเอง ผมจึงจัดรายการ New และกิจกรรม cmNewDraw ไว้ให้ในรูปที่ 9 เช่นกัน การ InvalidateRect ใช้เพื่อการเตรียมพื้นที่ในการวาดภาพ แต่ที่ผมนำมาใช้นี้เพื่อการลบภาพในหน้าต่างคือที่ให้ค่า true และฟังก์ชันนี้จะเป็นผู้เรียกใช้

```
(* ---------- WndProC  Method ---------- *)

CONSTRUCTOR WndProC.Init(apar : PWindowsObject; atitle : PChar);
BEGIN
 TWindow.Init(apar,atitle); ox := 20; oy := 20;
 attr.W := 260; attr.H := 200;
 attr.menu := LoadMenu(HInstance, mainmenu);
END;


PROCEDURE WndProc.GetWindowClass(var awndclass : TWndClass);
BEGIN
 TWindow.GetWindowClass(awndclass);
 awndclass.HIcon := LoadIcon(HInstance, iconame);
END;


PROCEDURE WndProC.Paint(paintdc : HDC;
                var paintInfo : TPaintStruct);
BEGIN
 TWindow.Paint(paintdc,paintinfo);
 ox := 20; oy := 20;
END;


PROCEDURE WndProc.cmReNew(var msg: TMessage);
BEGIN
 InvalidateRect(HWindow, nil, true);
END;
```

รูปที่ 9 โปรแกรมที่ใช้ในการลบภาพบนจอภาพ รวมทั้งโปรแกรมที่เหลือ

```
PROGRAM IconStock;

{$R icostock.res}

USES  WinTypes,WinProcs,WObjects,Strings;
CONST appname  = 'icostock';
      thetitle = 'ICON STOCK';
      mainmenu = 'ICON_MENU';
      iconame  = 'CIR_ICO';

      cm_Omenus = 1000;     cm_ReNew = 1001;

TYPE
 WndProCP = ^WndProC;
 WndProC = OBJECT(TWindow)
  ox,oy : Integer;
 constructor Init(apar : PWindowsObject; atitle : PChar);
 procedure  GetWindowClass(var awndclass : TWndClass); virtual;
 procedure  Paint(paintdc : HDC;
           var paintInfo : TPaintStruct); virtual;
 procedure  wmCommand(var Msg: TMessage);
           virtual wm_First + wm_Command;
 procedure  cmReNew(var Msg: TMessage);
           virtual cm_First + cm_ReNew;
 procedure  DispIcon(iconId : Integer);
        END;
```

รูปที่ 10 ตอนต้นของโปรแกรม IconStock และการกำหนดสกุล WndProC

Paint ด้วย ทำให้เกิดผลเช่นเดียวกับเมื่อเปลี่ยนขนาดของหน้าต่าง คือให้ผู้ใช้เริ่มต้นเลือกรายการใหม่

ให้แสดงภาพใหม่

แต่ถ้าต้องการให้ภาพไอคอนยังคงอยู่เมื่อมีการเปลี่ยนแปลงขนาดของหน้าต่างจะต้องเขียนโปรแกรมเพิ่มครับ กล่าวคือ

1. จัดที่เก็บข้อมูล (สกุล TCollection) ไว้เก็บรายการที่ผู้ใช้เลือก ซึ่งจะต้องพิจารณาว่าจะเก็บเป็นค่าไอดีใน wParm หรือเก็บเป็นค่าไอคอนรีซอร์สที่อ่านมาจากเมนูแล้ว ถ้าต้องการเขียนโปรแกรมให้สั้นแต่ทำงานได้ช้ากว่าคือเก็บค่าไอดี ซึ่งจะขอกล่าวตามนี้

2. ให้ wmCommand เก็บค่าไอดีลงที่เก็บ

3. ให้ Paint นำค่าไอดีในที่เก็บออกแสดงด้วย DispIcon

4. ให้ cmNewDraw ถอนข้อมูลออกจากที่เก็บทั้งหมด (Paint จึงไม่มีอะไรที่จะนำออกแสดง)

โปรแกรมทั้งหมด

ได้แสดงโปรแกรมตอนต้นไว้ในรูปที่ 10 การป้อนโปรแกรมขอให้นำส่วนที่เป็นโปรแกรมใหม่ในรูปที่ 10, 9 และ 8 ไปป้อนแทรกในรูปที่ 5 ครับ

บิตแม็พ

บิตแม็พมีหลายชนิด ไอคอนก็เป็นบิตแม็พชนิดหนึ่ง ที่จะกล่าวถึงนี้จะเป็นบิตแม็พที่ใช้แสดงภาพในความหมายที่เป็นภาพแท้ ๆ รวมทั้งนำไปใช้แทนชื่อรายการในเมนู สามารถสร้างภาพได้ 3 วิธีด้วยกันคือ

– ใช้บิตแม็พเอดิเตอร์ของ WRT วาดแล้วเก็บลงไฟล์ .BMP หรือผนวกเข้ากับไฟล์ .RES

– ใช้สแกนเนอร์อ่านเข้ามา หรือวาดด้วยเพนต์บรัชเก็บไว้ในไฟล์ .BMP (1 รูปต่อ 1 ไฟล์) เมื่อต้องการทำเป็นรีซอร์สต้องใช้บิตแม็พเอดิเตอร์ของ WRT อ่านเข้ามาผนวกเข้ากับไฟล์ .RES

– ด้วยการวาดภาพในโปรแกรม

สามารถจะโหลดบิตแม็พได้ทั้งจากไฟล์ .BMP และจากรีซอร์สในไฟล์ .RES โดยที่การโหลดจากรีซอร์สเขียนโปรแกรมได้สั้นกว่ากันมาก แต่ต้องเปลืองหน่วยความจำหลักเพราะส่วนมากจะโหลดตั้งแต่อินิต ส่วนการโหลดจากไฟล์ .BMP สามารถโหลดเมื่อต้องการใช้ได้ โดยทั่วไปจึง

– ใช้รีซอร์สเมื่อเป็นภาพขนาดเล็กหรือต้องใช้ตั้งแต่ต้นอยู่แล้ว

– โหลดจากไฟล์ .BMP เมื่อมีหลายภาพ (หลายไฟล์) ผลัดเปลี่ยนกันแสดงเช่น ในนิทรรศการ

ในบทความนี้จะกล่าวถึงการใช้รีซอร์ส ซึ่งสร้างด้วย WRT สำหรับการโหลดจากไฟล์โดยตรงขอให้ดูโปรแกรมในไฟล์ BSCRLAPP.PAS ของเทอร์โบปาสคาล (อยู่ในไดเรคทอรี C: \ TPW \ OWLDEMOS)

บิตแม็พเอดิเตอร์

เมื่อเลือกบิตแม็พเอดิเตอร์ คือเลือกปุ่ม Bitmap จากหน้าต่างหลักของ WRT จะแสดงภาพดังรูปที่ 11 ซึ่งผมได้โหลดไฟล์ RACECAR.BMP ในชุดของเทอร์โบปาสคาลมาแสดงด้วย หน้าต่างเหมือนกับของไอคอนครับ จึงขอกล่าวเฉพาะข้อแตกต่าง

ขนาดของภาพ

หากต้องการวาดภาพใหม่เลือกรายการ File/New จะแสดงไดอะล็อกบ็อกซ์ ให้ป้อนขนาดของภาพ ขนาดที่แนะนำไว้คือ 72×72 ซึ่งสามารถกำหนดได้ตั้ง 1×1 ไปจนถึงหน่วยความจำไม่พอครับ

แต่ถ้าโหลดภาพเก่าด้วย File/Open จะมีขนาดของภาพเท่ากับขนาดของบิตแม็พนั้น เช่นตามรูปนี้คือ 320×160 ซึ่งจะต้องจดไว้เพื่อใช้ในโปรแกรมต่อไป

เช่นเดียวกับไอคอนคือจะแสดงภาพใน 2 กรอบ กรอบทางขวามือคือกรอบเท่าของจริง ส่วนทางซ้ายคือกรอบขยายซึ่งผมให้ขยายไว้ 4 เท่า ถ้าภาพล้นกรอบสามารถเลื่อนดูได้ด้วยสโครลบาร์หรือมือเลื่อน

สี

จานสีคงมีสีเท่ากับของไอคอน แต่ไม่มีกรอบสีพื้นและสีกลับของจอภาพคงมีแต่สีคือ Color กรอบเดียว เพื่อแสดงว่าขณะนั้นเลือกสีใดอยู่ ที่เพิ่มมาคือรายการ Pallette ให้เราผสมสีให้แก่แต่ละกรอบสีได้ใหม่ โดยเลือกกรอบสีแล้วเลือกรายการนี้ มีรายการรองให้เลือกคือ

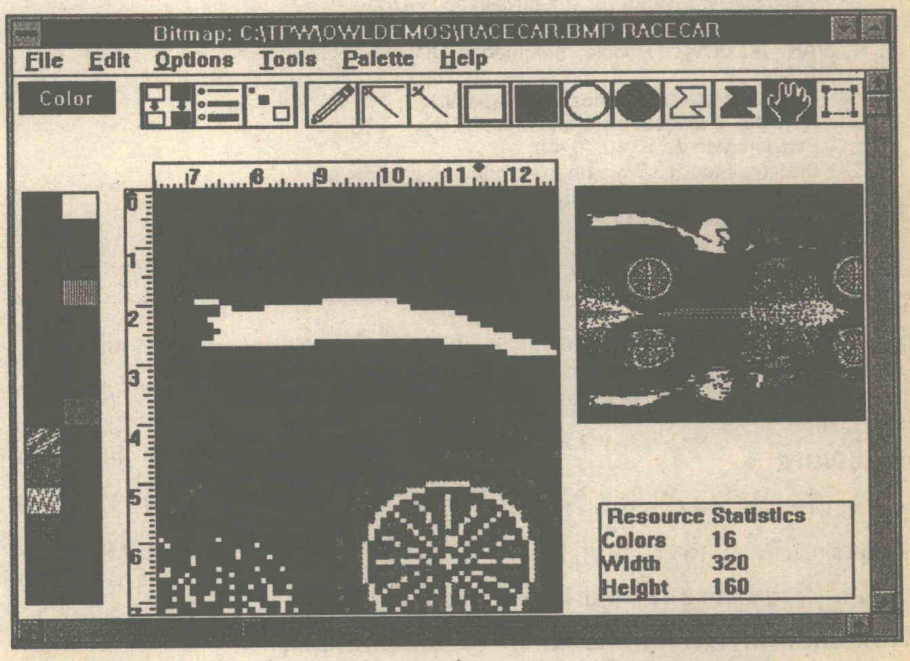
– Edit Color ให้ผสมสีใหม่ โดยใช้สโครลบาร์ เช่นเดียวกับในเพนต์บรัช

– Save Colors เก็บค่าในจานสีไว้ในไฟล์ เพื่อโหลดมาใช้ในภายหลังด้วย Get Colors

นอกจากนี้ก็มีรายการ Options/Combination เพื่อให้เลือกสีในการอ่านภาพจากคลิปบอร์ด

แสดงภาพบิตแม็พ

เมื่อสร้างภาพบิตแม็พแล้วซึ่งคุณอาจบรรจงวาดเองหรือผมก็อบปี้เขามาก็เพื่อนำมาใช้ในโปรแกรมครับ ได้แสดงโปรแกรม DemoBitmap ไว้ในรูปที่ 12 คงขาดส่วนของ TheAplC ตามเคย ตอนต้นของโปรแกรมได้ให้ค่าคงตัวต่าง ๆ ไว้ มีการดำเนินการต่าง ๆ คือ



รูปที่ 11 หน้าต่างบิตแม็พเอดิเตอร์ของ WRT หลังจากที่ได้โหลดไฟล์ RACECAR.BMP แล้ว

```
PROGRAM DemoBitmap;  { DEMOBMP.PAS }

{$R demobmp.res}

USES  WinTypes,WinProcs,WObjects;
CONST appname  = 'demobmp';
      thetitle = 'DEMO BITMAP';
      bmpname  = 'RACECAR_BMP';

TYPE
 WndProCP = ^WndProC;
 WndProC = OBJECT(TWindow)
   hCar  : HBitmap;
   ww,hh : Integer;
 constructor Init(apar : PWindowsObject; atitle : PChar);
 destructor  Done; virtual;
 procedure  Paint(paintdc : HDC;
           var paintInfo : TPaintStruct); virtual;
          END;

(* ---------- WndProC  Method ---------- *)

CONSTRUCTOR WndProC.Init(apar : PWindowsObject; atitle : PChar);
BEGIN
 TWindow.Init(apar,atitle);
 attr.x := 30;  attr.y := 20;
 attr.w := 600;  attr.h := 400;
 ww := 320;  hh := 160;  { 320,160 }
 hCar := LoadBitMap(hInstance, bmpname);
END;


DESTRUCTOR WndProC.Done;
BEGIN
 DeleteObject(hCar);  TWindow.Done;
END;


PROCEDURE  WndProC.Paint(paintdc : HDC;
                  var paintInfo : TPaintStruct);
VAR  hOldObj : THandle;  hMemdc : HDC;
BEGIN
 hMemdc := CreateCompatibleDC(paintdc);
 hOldObj := SelectObject(hMemdc, hCar);
 Ellipse(hMemdc,10,10,40,40);
 BitBlt(paintdc, 10,5,ww,hh, hMemdc,0,0, srcCopy);
 SelectObject(hMemdc, hOldObj);
 DeleteDC(hMemDC);
END;
```

รูปที่ 12 โปรแกรม DemoBitmap เว้นโปรแกรมของสกุล TheAppC ซึ่งต้องลอกมาจากของเก่าเช่นจากรูปที่ 11

โหลดบิตแม็พ

จะเห็นว่าได้โหลดบิตแม็พตั้งแต่ต้นคือหลังจากอินิตหน้าต่าง โดย LoadBitmap ด้วยชื่อรีซอร์สของบิตแม็พเท่านี้ครับก็เป็นการโหลดบิตแม็พให้แก่ตัวแปร hCar ด้วย

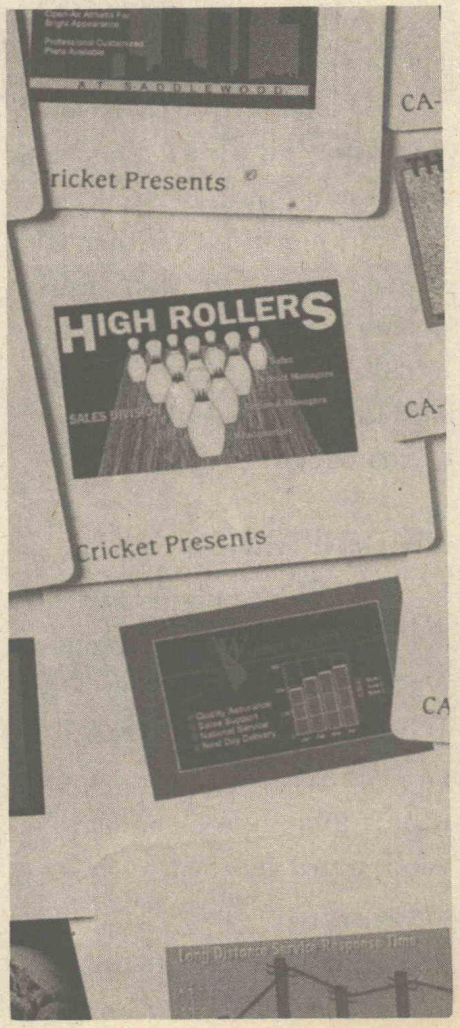
ค่าที่ให้แก่ ww และ hh คือค่าความกว้างและความสูงของบิตแม็พตามที่จดไว้ที่ทำให้เป็นตัวแปรเพื่อการเปลี่ยนขนาดดังจะกล่าวต่อไป

ยกเลิกบิตแม็พ

ยังไม่ทันนำบิตแม็พไปใช้เลยครับผมกลับยกเลิก ก็เพียงต้องการกล่าวก่อนเพราะเป็นเรื่องสำคัญมากเรื่องหนึ่งคือ เมื่อโหลดบิตแม็พลงหน่วยความจำจะครองพื้นที่ส่วนหนึ่งของหน่วยความจำไว้และมากเสียด้วย

ซึ่งเมื่อไม่ใช้งานบิตแม็พแล้วควรจะปลดปล่อยหน่วยความจำนี้ โดยเฉพาะเมื่อยุติการรันโปรแกรมนี้คือเมื่อผู้ใช้ปิดหน้าต่าง ไม่เช่นนั้นหน่วยความจำนี้จะถูกจองค้างอยู่ไม่มีใครนำไปใช้ได้จนกว่าจะยุติการรันวินโดวส์

จึงให้ยกเลิกบิตแม็พด้วย DeleteOpject ใน Done ซึ่งจะมาทำงานที่กิจกรรมนี้เมื่อผู้ใช้ปิดหน้าต่าง



ดิสเพลย์คอนเท็กซ์

ก่อนที่จะเดินเรื่องต่อไปขอนำเรื่องของดิสเพลย์คอนเท็กซ์มากล่าวถึงก่อน เพราะห่างเหินกันมานาน

ต้องขอย้อนกลับไปที่ดอส เช่น ถ้าเป็นเครื่องที่ใช้การ์ด VGA ก็จะมีหน่วยความจำหน่วยหนึ่งอยู่ที่แอดเดรส 8A000:000 เป็นต้นไป เป็นหน่วยความจำที่ถ้าเรานำภาพใดไปให้แก่หน่วยความจำนี้ จะปรากฏภาพนั้นบนจอภาพ จึงเรียกว่าหน่วยความจำจอภาพ (screen memory)

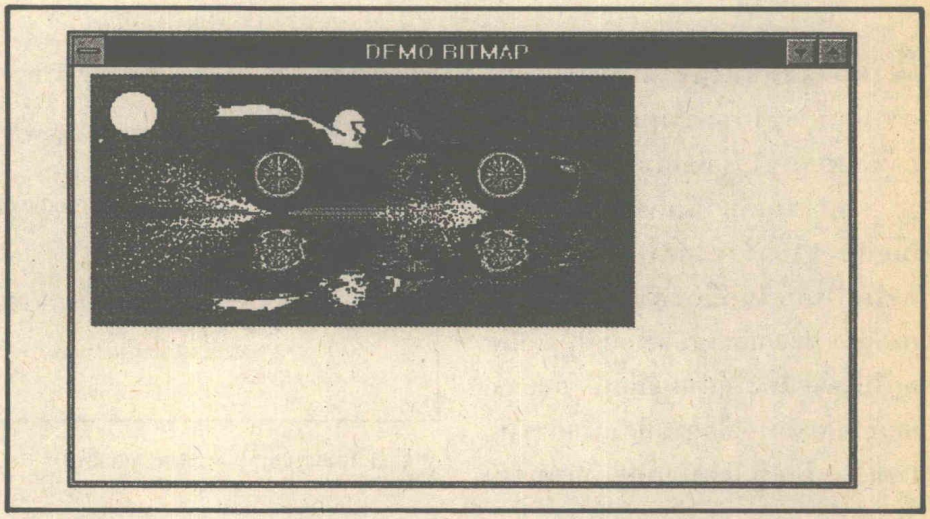
การจะนำไปไว้ได้อย่างไรทำให้เกิดภาพตรงไหนเป็นเรื่องยาวครับ ซึ่งต้องเขียนโปรแกรมให้และกับการ์ดจอภาพแต่ละชนิดด้วย หากเขียนโปรแกรมด้วยภาษาปาสคาลอาจไม่รู้สึก เพราะเทอร์โบปาสคาลจะติดตั้งตัวเองกับจอภาพนานาชนิดอยู่แล้ว นอกจากต้องการติดต่อกับจอภาพด้วยตนเอง เช่น การแสดงผลภาษาไทย

เมื่อมาถึงวินโดวส์ต้องการให้ผู้เขียนโปรแกรมเป็นอิสระจากจอภาพ และไม่ใช้แต่จอภาพอย่างเดียวยังรวมถึงเครื่องพิมพ์ชนิดต่าง ๆ และอุปกรณ์อื่นอีกด้วย ในวินโดวส์จึงไม่ให้เอาต์พุตไปที่อุปกรณ์โดยตรง หากให้เอาต์พุตไปที่หน่วยความจำหน่วยหนึ่ง เรียกว่าดิสเพลย์คอนเท็กซ์ (display context) หรือเรียกสั้น ๆ ว่า ดีซี (DC) เป็นหน่วยความจำแทนจอภาพ

ซึ่งจะต้องมีดีไวซ์ไดรเวอร์ของอุปกรณ์แต่ละชนิดแต่ละแบบ นำไปออกยังอุปกรณ์นั้นอีกทอดหนึ่ง เช่นถ้าเป็นจอภาพก็จะนำไปให้แก่หน่วยความจำจอภาพ

แต่พอมาถึงเรื่องของบิตแม็พ ในวินโดวส์ได้ให้ดำเนินการลึกลงไปอีกชั้นหนึ่งคือให้กำหนดหน่วยความจำอีกหน่วยหนึ่งเรียกว่า memory display context ซึ่งผมจะขอเรียกสั้น ๆ ว่า เมมดีซี เป็นหน่วยความจำเทียมจอภาพ เมื่อรวมแล้วก็หลายชั้นครับ เพราะให้เราดำเนินการดังนี้

- โหลดบิตแม็พลงหน่วยความจำหน่วยหนึ่งตามตัวอย่างนี้คือ hCar
- การกำหนดเมมดีซี แล้วให้บิตแม็พแก่เมมดีซี ทำให้สามารถปรับแต่งหรือต่อเติมภาพโดยผู้ใช้ไม่เห็น เพราะเมมดีซีมีฐานะเท่ากับดีซีเพียงแต่ยังไม่ได้ติดต่อกับไดรเวอร์
- ให้ค่าเมมดีซีแก่ดีซี จะทำให้เกิดภาพบนจอภาพ



รูปที่ 13 ผลการรันของโปรแกรม DemoBitmap

แสดงภาพ

ผมพร้อมที่จะแสดงภาพแล้วครับ ดังได้เขียนโปรแกรมไว้ใน Paint ซึ่งจะมาทำงานที่กิจกรรมนี้เมื่อเปิดหน้าต่าง และทุกครั้งที่ผู้ใช้ย่อหรือขยายภาพของหน้าต่าง โดยได้มีการดำเนินการดังนี้คือ

1. กำหนดเมมดีซี ให้สอดคล้องกับดีซีคือ paintdc ด้วย CreateCompatibleDC

2. ให้บิตแม็พแก่เมมดีซีด้วย SelectObject ซึ่งจะให้ค่าแฮนเดิลเดิมคืนมาผมให้เก็บไว้ใน hOldObj (มีเรื่องที่จะกล่าวถึงต่อไป) ตรงนี้ครับที่ทำให้เราแก้หรือต่อเติมภาพได้ ดังที่ผมให้ทำเป็นรูปวงกลม

3. ให้ค่าในเมมดีซีแก่ดีซีด้วย BitBlt (Blt ย่อมาจาก Blasting) ค่าพารามิเตอร์ที่ผ่านไปให้คือ

- paintdc คือดิสเพลย์คอนเท็กซ์
- ค่า 10, 5 คือตำแหน่งมุมบนซ้ายของภาพในหน้าต่าง
- ค่า ww และ hh คือความกว้างและความสูงของภาพ
- hMemdc คือเมมดีซี
- ค่า 0, 0 คือตำแหน่งมุมบนซ้ายที่จะให้เริ่มต้นลอกภาพ เช่น ถ้าให้เป็น 160, 80 ก็จะได้ภาพเฉพาะทางด้านล่างขวา
- scrCopy เป็นแฟล็กหมายถึงให้ก๊อบปี้ไปตรง ๆ มีค่าอื่นอีกขอให้ดูในฟังก์ชัน BitBlt ซึ่งใน Help จะให้คำอธิบายไว้มากกว่า

จากผลของโปรแกรมมาถึงจุดนี้จะปรากฏภาพบนจอภาพดังรูปที่ 13

(ซึ่งผมได้ลดขนาดของหน้าต่างเพื่อความเหมาะสมในการพิมพ์) จะเห็นภาพวงกลมขาวที่ต่อเติมในโปรแกรม

4. เช่นเดียวกับดีซีทั้งหลายคือเมื่อหมดธุระแล้วจะต้องยกเลิกตามวิธีการที่กำหนดในคู่มือของไมโครซอฟต์ กล่าวว่า จะยกเลิกเมมดีซีโดยมีบิตแม็พอยู่ไม่ได้จะต้องให้ค่าเดิมก่อน และจะยกเลิกบิตแม็พ (คือตามที่ดำเนินการใน Done) โดยที่ยังผนวกอยู่กับเมมดีซีไม่ได้ จึง

– ให้ค่าเดิมที่เก็บไว้ใน hOldObj แก่ hMemdc ด้วย SelectObj ซึ่งค่าที่ SelectObj จะส่งกลับมาคือค่าของ hCar จึงไม่ต้องเก็บไว้

– ยกเลิกเมมดีซีด้วย DeleteDC (ทำให้ใช้คำสั่ง DeleteObj ใน Done ได้ด้วย)

ย่อและขยายภาพ

ภาพที่ลอกจากเมมดีซีไปให้แก่ดีซีสามารถให้ลอกในแบบย่อหรือขยายได้ โดยใช้ฟังก์ชัน StretchBit แทน BitBlt ดังแสดงในรูปที่ 14 ค่าพารามิเตอร์ที่เพิ่มมาคือ 320 และ 160 คือความกว้างและความสูงของภาพเดิม ทำให้สามารถลอกเพียงบางส่วนของจอภาพเดิมได้

จะเป็นการย่อหรือขยาย ถ้ากำหนดให้ค่า ww x hh น้อยกว่าภาพเดิมเช่นกรณีนี้คือ 320×160 ก็จะเป็นการย่อ ถ้ามากกว่าก็จะเป็นการขยาย ซึ่งจะเป็นสัดส่วนของแต่ละด้านด้วย

เพื่อให้เห็นภาพขนาดต่าง ๆ ต่อเนื่องกัน จึงได้กำหนดกิจกรรม wmRButtonDown เพิ่มด้านทั้งสองขึ้นเท่าตัวทุกครั้งที่ผู้ใช้คลิ๊กปุ่มขวาของเม้าส์ โดยได้ให้ค่า ww และ hh ใน Init ใหม่เป็น 40 และ 20 ตามลำดับ เพื่อให้เห็นการย่อภาพ

```
StretchBlt(paintdc, 10,5,ww,hh, hMemdc,0,0, 320,160, srcCopy);


PROCEDURE WndProC.wmRButtonDown(var Msg: TMessage);
BEGIN
 ww := ww * 2;
 if ww > 320*3 then
  begin ww := 160; hh := 80 end
 else hh := hh * 2;
 InvalidateRect(HWindow, nil, True);
END;
```

รูปที่ 14 โปรแกรมที่ใช้ในการย่อและขยายภาพ

เคอร์เซอร์

ในวินโดวส์เรียกพอยน์เตอร์ของเม้าส์ว่า เคอร์เซอร์ (cursor) ส่วนที่เรียกว่าเคอร์เซอร์เดิมบนดอสกลับเรียกว่า แคเรต (caret) ทำไมถึงต้องเรียกเช่นนี้ไม่ทราบเหตุผลครับ แต่ก็ต้องเรียกตามไม่เช่นนั้นคงอ้างไม่ถึงกัน

ในแอพพลิเคชันคือโปรแกรมเช่นที่ผ่านมา ถึงแม้เราจะไม่โหลดเคอร์เซอร์ก็มีเคอร์เซอร์ใช้ เพราะเทอร์โบปาสคาลดำเนินการแทนเราใน TWindow โดยใช้เคอร์เซอร์ ของระบบที่เป็นรูปลูกศรเอียง และถ้าเราไม่สร้างเคอร์เซอร์ของเราเองก็สามารถโหลดเคอร์เซอร์ของระบบรูปแบบอื่นเช่นเป็นรูปกากบาทมาใช้ได้

โดยที่ในวินโดวส์ได้วางระบบไว้ว่าเมื่อผู้ใช้เลื่อนเคอร์เซอร์เข้าไปในพื้นที่ที่ใช้งานของหน้าต่างใด ภาพเคอร์เซอร์จะเปลี่ยนเป็นของหน้าต่างนั้นไม่ว่าหน้าต่างนั้นจะแอคตีฟหรือไม่ ผมกล่าวว่าพื้นที่ใช้งานคือไม่รวมกรอบไตเติ้ลบาร์ เมนูบาร์ และอื่น ๆ ที่มีเคอร์เซอร์ของตนเองอยู่แล้ว เช่น กรอบหน้าต่างมีเคอร์เซอร์เป็นลูกศรสองทิศทาง

สร้างเคอร์เซอร์รีซอร์ส

เมื่อต้องการใช้เคอร์เซอร์ของเราเองก็ต้องสร้างรีซอร์สด้วย WRT อีกเช่นกัน โดยเลือกเคอร์เซอร์เอดิเตอร์ คือปุ่ม Cursor จะแสดงหน้าต่างของเอดิเตอร์นี้ เหมือนกับของไอคอน

ข้อแตกต่างคือทางขวาสุดของแถวปุ่มเครื่องมือจะมีปุ่มฮอตสปอต (Hot-spot) ซึ่งต้องขยายหน้าต่างจึงจะเห็น ปุ่มนี้สำคัญมากลืมไม่ได้ เพื่อกำหนดว่า

ต้นฉบับไม่มีหน้า 333

No This Page 333 In Original